Национальный исследовательский университет «Высшая школа экономики»

Факультет **«Бизнес-информатики»**
Отделение программной инженерии

Кафедра
Управление разработкой программного обеспечения

Контрольное домашнее задание
по дисциплине
«Программирование»

Номер варианта работы: 2 вариант

Выполнил: Студент группы 171ПИ (1)
_____ Сероусов В.Э.
тел. _____
e-mail адрес: _____

Преподаватель: _____

Москва, 2014 год. Модуль № 3

Оглавление

1. Условие задачи

Вариант 2.

Определить базовый класс «функция одного аргумента» с абстрактными методами: для вычисления значения функции при заданном значении аргумента; для задания параметров функции; для представления записи функции (в текстовом виде).

Производные классы: «линейная функция»; «параболическая функция»; «гиперболическая функция». В производных классах должны быть переопределены все методы базового класса и при необходимости добавлены новые методы.

Определить контейнер (с элементами, отличными от типа object) для хранения объектов разных производных классов. Создать несколько объектов производных классов и поместить их в контейнер.

Приложение должно обеспечивать пользователю возможности:

- добавлять в контейнер новую функцию (новый объект)
- удалять из контейнера выбранную пользователем функцию
- выводить в виде таблицы сведения обо всех функциях из контейнера
- сохранять объекты контейнера в текстовом файле
- читать записи из файла в контейнер
- упорядочивать функции по значениям при заданном пользователем значении аргумента
- выводить список всех функций заданного вида

2. Функции разрабатываемого приложения

2.1. Варианты использования

Приложение может использовано для хранения и сортировки линейных, параболических и гиперболических функций. Приложение работает с текстовыми файлами формата txt записанных в кодировке UTF-8.

2.2. Описание интерфейса пользователя

Приложение содержит 3 элемента типа button, 4 элемента типа textBox, 3 элемента типа checkBox, 1 элемент типа comboBox, 4 элемента типа label, 2 элемента типа groupBox и 1 элемент типа tableLayoutPanel (контейнеры для объединения элементов), 1 элемент listView (для отображения данных).

Программа содержит так же такие элементы управления как menuStrip, toolStrip (меню и кнопки), toolTip (всплывающие подсказки), openFileDialog (открытие файла), saveFileDialog (закрытие файла).

Начальная настройка элементов управления:

Form1: MaximizeBox = false, MinimizeBox = false, FormBorderStyle = “FixedSingle”

listView1: BorderStyle = “Fixed3D”, HeaderStyle = “Clickable”, View = “Details”, Scrollable = true, MultiSelect = false, FullRowSelect = true

button1: Text = “Добавить функцию”

button2: Text = “Очистить форму”

button3: Text = “Удалить выбранное”, Enabled = false

Меню позволяет выполнять следующие функции:

- Создание файла
- Открытие файла
- Сохранение файла с исходным именем
- Сохранение файла с заданным именем
- Просмотр сведений “О программе”
- Выход из программы

В заголовке формы отображается название файла, открытого в данный момент.

С помощью элемента comboBox1 пользователь может выбрать тип математической функции. Поля “аргумент функции” и “коэффициенты (параметры)" являются обязательными для заполнения и являются основными для вычисления значения функции.

Сероусов Виталий. Вариант 2

Кнопка “Добавить функцию” позволяет добавить в список функцию с заданным типом, аргументом и коэффициентами (параметрами).

Кнопка “Очистить форму” позволяет очистить поля заполнения аргумента и коэффициентов (параметров), а также сбросить поле типа функции.

Элементы управления типа checkbox с надписями “Линейные”, “Параболические” и “Гиперболические” позволяют включать\отключать отображения в списке функций того или иного типа.

Для удаления функции из списка следует выделить её и нажать на кнопку с надписью “Удалить выбранное”.

Наиболее часто используемые пункты меню расположены ниже с помощью элемента ToolStrip.

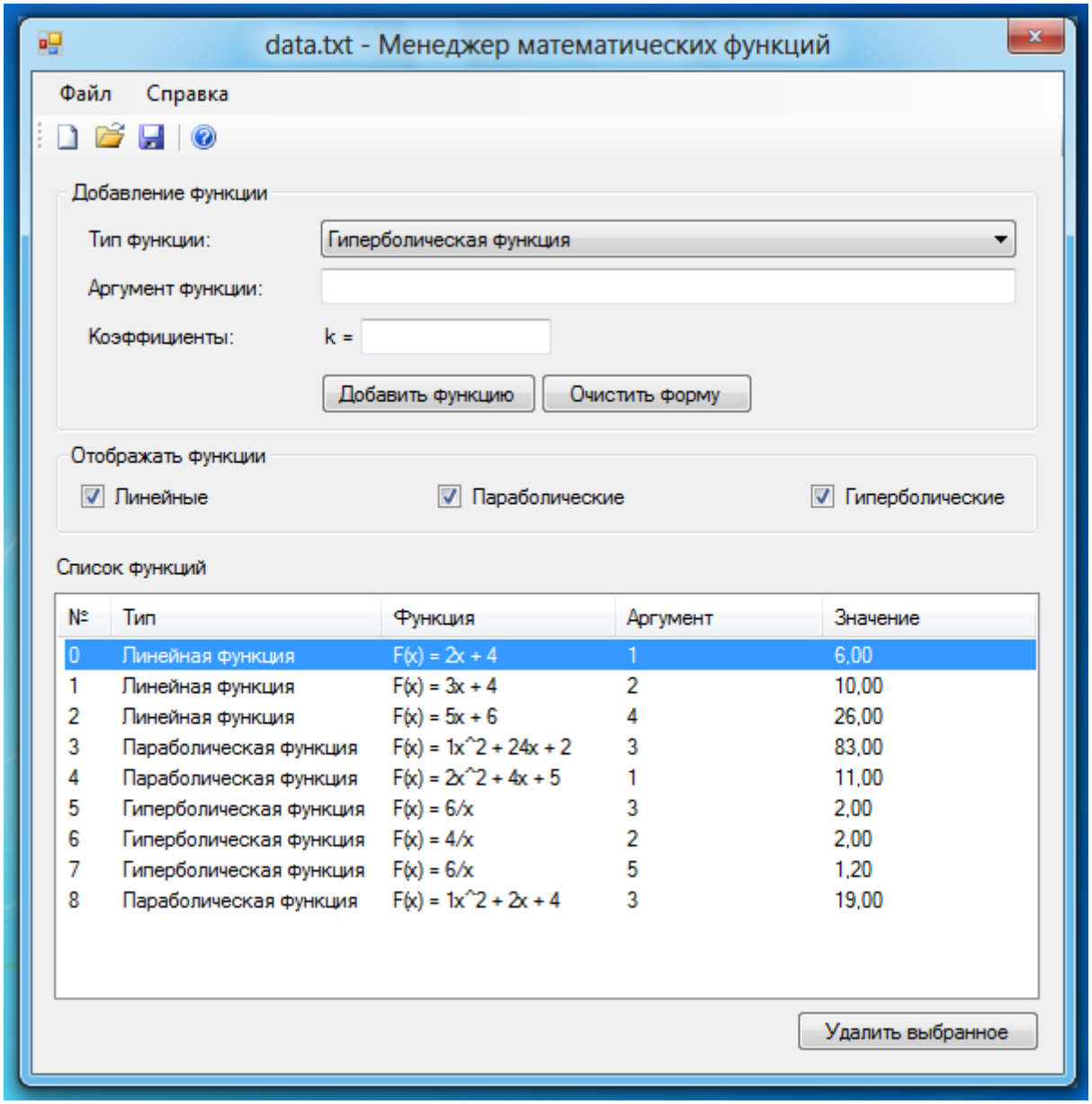

Рис.1. Скриншот интерфейса программы

3. Структура приложения

3.1. Диграмма классов

Диаграмма классов библиотеки MathFunctions

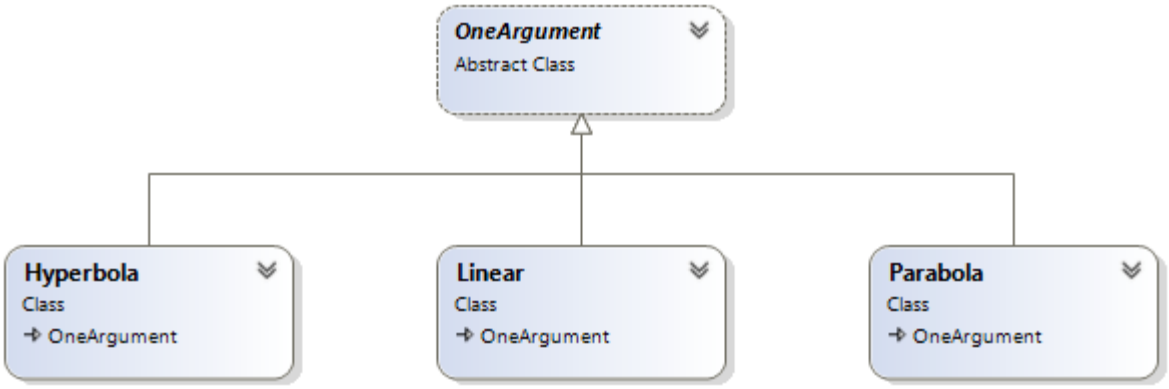

Рис.2. Диаграмма классов в пространстве имен MathFunctions

Диаграмма классов в пространстве имен WindowsFormsApplication

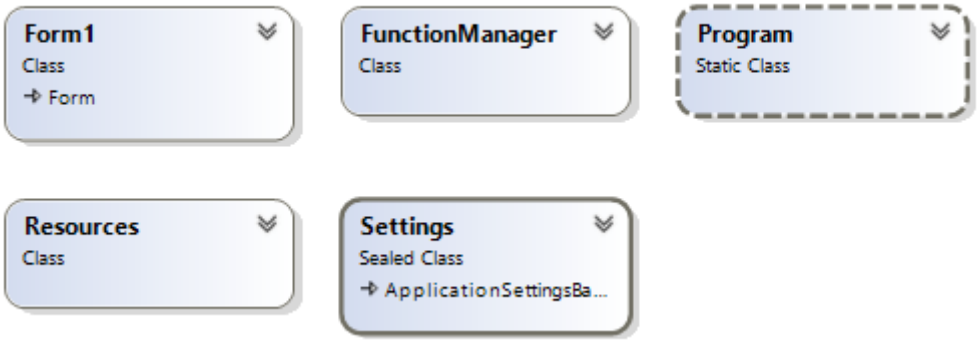

Рис.3. Диаграмма классов в пространстве имен WindowsFormsApplication

3.2. Описание классов их полей, методов

Библиотека **`MathFunctions`**:

Абстрактный базовый класс **`OneArgument`** содержит в себе следующие авто реализуемые свойства:

- `string name` – содержит название типа функции
- `double argument` – содержит аргумент функции
- `double value` – содержит значение функции

Следующие абстрактные методы:

- `double GetParam(int order)` – возвращает значение параметра (коэффициента)
- `void Calculate(double x)` – производит вычисление значения функции при заданном аргументе
- `void Parameters(params double[] coefficients)` – задаёт коэффициенты функции (параметры)
- `void Show()` – выводит на экран представление записи функции

Класс **`Linear`** унаследован от класса **`OneArgument`** и содержит реализацию всех абстрактных методов класса **`OneArgument`**. Кроме того, класс **`Linear`** имеет конструктор без параметров, задающий имя типа функции (в данном случае “Линейная функция”). Класс **`Linear`** также содержит следующие приватные поля:

- `double k` – коэффициент k линейной функции
- `double b` – коэффициент b линейной функции

Класс **`Hyperbola`** унаследован от класса **`OneArgument`** и содержит реализацию всех абстрактных методов класса **`OneArgument`**. Кроме того, класс **`Hyperbola`** имеет конструктор без параметров, задающий имя типа функции (в данном случае “Гиперболическая функция”). Класс **`Hyperbola`** также содержит следующее приватное поле:

- `double k` – коэффициент k гиперболической функции

Класс **`Parabola`** унаследован от класса **`OneArgument`** и содержит реализацию всех абстрактных методов класса **`OneArgument`**. Кроме того, класс **`Parabola`** имеет конструктор без параметров, задающий имя типа функции (в данном случае “Параболическая функция”). Класс **`Parabola`** также содержит следующие приватные поля:

- `double a` – коэффициент a параболической функции
- `double b` – коэффициент b параболической функции
- `double c` – коэффициент c параболической функции

WindowsForms-приложение **`WindowsFormsApplication`**:

Класс **`FunctionManager`** содержит открытое поле списка **`MathFunctions`** с типом **`OneArgument`** (`List<OneArgument> MathFunctions`) для хранения объектов, типы которых унаследованы от класса **`OneArgument`,** а именно **`Linear`**, **`Parabola`**, **`Hyperbola`.** Класс **`FunctionManager`** также содержит следующие открытые методы:

- `void CreateFile(string filename)` – создаёт файл формата txt с именем filename и создаёт новый экземпляр для ссылки `MathFunctions`
- `void OpenFile(string filename)` – открывает файл формата txt с именем filename, а также строит список объектов по данным прочитанным из файла
- `void SaveFile(string filename)` – сохраняет данные, хранящиеся в списке `MathFunctions` в файле filename формата txt, который в исходный момент открыт.
- `void SaveFileAs(string filename)` – сохраняет данные, хранящиеся в списке `MathFunctions` в файле filename формата txt с именем отличным от исходного файла.
- `int getNumParams(int index)` – возвращает число параметров (коэффициентов) функции в зависимости от её типа.
- `void SortByValue()` – сортирует по значению в порядке возрастания, если не отстортирована функция или отсортирована в порядке убывания), иначе сортируется в порядке убывания.

Класс **`Form1`** содержит методы являющиеся обработчиками событий элементов формы. Кроме того, он содержит следующие константы:

- `int numTypes = 3` – количество типов математических функций
- `int defaultIndex = 0` – умалчиваемое значение comboBox
- `string defaultFirstK = "k = "` – умалчиваемое значение label первого коэффициента (параметра)
- `string defaultSecondK = "b = "` – умалчиваемое значение label второго коэффициента
- `string captionError = "Ошибка!"` – текст ошибки, выводимый в окнах MessageBox
- `string messageError = "Заполнены не все поля или заполнены неверно!"` – ошибка, всплывающая в окне MessageBox в случае, если произойдёт попытка добавления функции при незаполненных или неверно заполненных полях.
- `string title = "Менеджер математических функций"` – название программы

А также следующее поле:

- `string currentFile = "./data.txt"` – текущий открытый файл, по умолчанию “./data.txt”

Следующие методы (за исключением методов, обрабатывающих события):

- `void resetForm()` – сбрасывает поля типа, аргумента и коэффициентов в форме
- `void updateListView()` – обновляет отображение данных в `listView()`, читая данные из контейнера с учетом включение\выключения отображения тех или иных типов, отмеченных в checkbox

4. Распределение исходного кода по файлам проекта

- OneArgument.cs - содержится базовый абстрактный класс `OneArgument`, отвечающий за структуру классов унаследованных от него.
- Parabola.cs – содержится класс `Parabola`, унаследованный от класса `OneArgument`, позволяющий производить вычисления для параболической функции.
- Linear.cs - содержится класс `Linear`, унаследованный от класса `OneArgument`, позволяющий производить вычисления для линейной функции.
- Hyperbola.cs – содежрится класс `Hyperbola`, унаследованный от класса OneArgument, позволяющий производить вычисления для гиперболической функции
- Form1.cs – содержится класс `Form1`, отвечающий за обработку событий формы программы
- FunctionManager.cs – содержится класс `FunctionManager`, отвечающий за управление контейнером объектов, производных от типа `OneArgument`.
- Program.cs – точка входа в программу, создана средой автоматически.

5. Контрольный пример и описание результатов

Контрольный пример хранится в файле формата txt в каталоге программы (в папке debug) с именем data.txt, в котором для примера добавлено 8 математических функций 3-х видов (линейная, параболическая, гиперболическая). Было протестировано удаление функций, добавление в файл новых, а также их сортировка. Результаты хранятся в файле data2.txt.

6. Текст (код) программы

Файл **OneArgument.cs:**

```
namespace MathFunctions
{
    public abstract class OneArgument
    {
        public string name
        {
            protected set;
            get;
        }

        public double argument
        {
            protected set;
            get;
        }

        public double value
        {
            protected set;
            get;
        }

        public abstract double GetParam(int order);
        public abstract void Calculate(double x);
        public abstract void Parameters(params double[] coefficients);
        public abstract string Show();
    }
}
```

Файл **Hyperbola.cs:**

```
namespace MathFunctions
{
    public class Hyperbola : OneArgument
    {
        private double k;

        public Hyperbola()
        {
            base.name = "Гиперболическая функция";
        }

        public override double GetParam(int order)
        {
            if(order > 0)
            {
                throw new Exception("Такого параметра нет");
            }
            return k;
        }


       public override void Calculate(double x)
        {
            if (x == 0)
            {
                throw new Exception("Делить на ноль нельзя");
            }
            base.argument = x;
            base.value = k / x;
        }
        public override void Parameters(params double[] coefficients)
```

```
        {
            int length = coefficients.Length;

            if (length > 3 || length <= 0)
            {
                throw new Exception("Неверное число параметров!");
            }

          if(coefficients[0] == 0)
          {
            throw new Exception("Функция не является гиперболой при k = 0");
          }

            k = coefficients[0];
        }
        public override string Show()
        {
            return "F(x) = " + k + "/x";
        }
    }
}
```

Файл **Parabola.cs:**

```
namespace MathFunctions
{
    public class Parabola : OneArgument
    {
        private double a;
        private double b;
        private double c;

        public Parabola()
        {
            base.name = "Параболическая функция";
        }

        public override double GetParam(int order)
        {
            if (order > 2)
            {
                throw new Exception("Такого параметра нет");
            }

            if (order == 0)
            {
                return a;
            }
            else if(order == 1)
            {
                return b;
            }
            else
            {
                return c;
            }
        }

        public override void Calculate(double x)
        {
            base.argument = x;
            base.value = a * x * x + b * x + c;
        }
        public override void Parameters(params double[] coefficients)
        {
            int length = coefficients.Length;

            if (length > 3 || length <= 0)
            {
                throw new Exception("Неверное число параметров!");
```

```
            }

            a = coefficients[0];
            b = length > 1 ? coefficients[1] : 0;
            c = length > 2 ? coefficients[2] : 0;
        }
        public override string Show()
        {
            if (a == 0 && b == 0 && c == 0)
            {
                return "F(x) = 0";
            }
            return "F(x) = " + (a != 0 ? a + "x^2 " : "")
                             + (b != 0 ? (b > 0 ? "+ " : "") + b.ToString().Replace("-
", "- ") + "x " : "")
                             + (c != 0 ? (c > 0 ? "+ " : "") + c.ToString().Replace("-
", "- ") : "");
        }
    }
}
```

Файл **Linear.cs:**

```
namespace MathFunctions
{
    public class Linear : OneArgument
    {
        private double k;
        private double b;
        public Linear()
        {
            base.name = "Линейная функция";
        }
        public override double GetParam(int order)
        {
            if (order > 1)
            {
                throw new Exception("Такого параметра нет");
            }

            if(order == 0)
            {
                return k;
            }
            else
            {
                return b;
            }
        }
        public override void Calculate(double x)
        {
            base.argument = x;
            base.value = k * x + b;
        }
        public override void Parameters(params double[] coefficients)
        {
            int length = coefficients.Length;

            if (length > 3 || length <= 0)
            {
                throw new Exception("Неверное число параметров!");
            }

            k = coefficients[0];
            b = length == 2 ? coefficients[1] : 0;
        }
        public override string Show()
        {
            if(k == 0 && b == 0)
            {
```

```
                return "F(x) = 0";
            }
            return "F(x) = " + (k != 0 ? k + "x " : "")
            + (b != 0 ? (b > 0 ? "+ " : "") + b.ToString().Replace("-", "- ") : "");
        }
    }
}
```

Файл **FunctionManager.cs:**

```
using MathFunctions;

namespace WindowsFormsApplication
{
    public class FunctionManager
    {
        public List<OneArgument> MathFunctions = new List<OneArgument>();

        public void CreateFile(string filename)
        {
            MathFunctions = new List<OneArgument>();
            FileStream file = new FileStream(filename, FileMode.Create,
FileAccess.Write, FileShare.None);
            file.Close();
        }

        public void OpenFile(string filename)
        {
            if(!File.Exists(filename))
            {
                throw new Exception("Файла не существует");
            }

            MathFunctions = new List<OneArgument>();
            StreamReader reader = new StreamReader(filename);
            string line;

            while ((line = reader.ReadLine()) != null)
            {
                string[] fields = line.Split('|');
                string type = fields[0];
                double argument = double.Parse(fields[1]);

                OneArgument function;

                if(type == "Параболическая функция")
                {
                    function = new Parabola();
                    double a = double.Parse(fields[2]);
                    double b = double.Parse(fields[3]);
                    double c = double.Parse(fields[4]);
                    function.Parameters(a, b, c);
                    function.Calculate(argument);
                }
                else if(type == "Линейная функция")
                {
                    function = new Linear();
                    double k = double.Parse(fields[2]);
                    double b = double.Parse(fields[3]);
                    function.Parameters(k, b);
                    function.Calculate(argument);
                }
                else
                {
                    function = new Hyperbola();
                    double k = double.Parse(fields[2]);
                    function.Parameters(k);
                    function.Calculate(argument);
                }
```

```
                MathFunctions.Add(function);
            }
            reader.Close();
        }

        public void SaveFile(string filename)
        {
            int count = MathFunctions.Count;

            StreamWriter writer = new StreamWriter(filename);

            for(int i = 0; i < count; i++)
            {
                OneArgument function = MathFunctions[i];
                string type = function.name;
                string argument = function.argument.ToString();
                if (function is Parabola)
                {
                    string a = function.GetParam(0).ToString();
                    string b = function.GetParam(1).ToString();
                    string c = function.GetParam(2).ToString();
                writer.WriteLine(type + "|" + argument + "|" + a + "|" + b + "|" + c);
                }
                else if(function is Linear)
                {
                    string k = function.GetParam(0).ToString();
                    string b = function.GetParam(1).ToString();
                    writer.WriteLine(type + "|" + argument + "|" + k + "|" + b);
                }
                else
                {
                    string k = function.GetParam(0).ToString();
                    writer.WriteLine(type + "|" + argument + "|" + k);
                }
            }
            writer.Close();
        }

        public void SaveFileAs(string filename)
        {
            FileStream file = new FileStream(filename, FileMode.Create,
FileAccess.Write, FileShare.None);
            file.Close();
            SaveFile(filename);
        }

        public int getNumParams(int index)
        {
            switch (index)
            {
                case 0:
                    return 2;
                case 1:
                    return 3;
                default:
                    return 1;
            }
        }
        public void SortByValue()
        {
            //Проверяем отсортирована ли коллекция
            bool isSortedAsc = MathFunctions.SequenceEqual(MathFunctions.OrderBy(x =>
x.value));
            if (!isSortedAsc)
               MathFunctions = MathFunctions.OrderBy(x => x.value).ToList();
            else
     MathFunctions = MathFunctions.OrderByDescending(x => x.value).ToList();
        }
    }
}
```

Файл **Form1.cs:**

```
using MathFunctions;

namespace WindowsFormsApplication
{
    public partial class Form1 : Form
    {
        FunctionManager manager;
        const int numTypes = 3;
        const int defaultIndex = 0;
        const string defaultFirstK = "k =";
        const string defaultSecondK = "b = ";
        const string captionError = "Ошибка!";
        const string messageError = "Заполнены не все поля или заполнены неверно!";
        const string title = "Менеджер математических функций";
        private string currentFile = "./data.txt";
        public Form1()
        {
            InitializeComponent();
            string defaultFile = Path.GetFileName(currentFile);
            manager = new FunctionManager();
            if (!File.Exists(currentFile))
            {
                manager.CreateFile(currentFile);
            }
            else
            {
                manager.OpenFile(currentFile);
            }
            this.Text = defaultFile + " - " + title;
            comboBox1.SelectedIndex = defaultIndex;
            label3.Text = defaultFirstK;
            label4.Text = defaultSecondK;
            label5.Text = "";
            textBox3.Visible = false;
            UpdateListView();
        }
        /// <summary>
        /// Очищаем форму
        /// </summary>
        private void resetForm()
        {
            int index = comboBox1.SelectedIndex;
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            if (index == 0)
            {
                label4.Visible = true;
                label5.Visible = false;
                label3.Text = "k =";
                label4.Text = "b =";
                label5.Text = "";
                textBox2.Visible = true;
                textBox3.Visible = false;
            }
            if (index == 1)
            {
                label4.Visible = true;
                label5.Visible = true;
                label3.Text = "a =";
                label4.Text = "b =";
                label5.Text = "c = ";
                textBox2.Visible = true;
                textBox3.Visible = true;
            }
```

```
            if (index == 2)
            {
                label4.Visible = false;
                label5.Visible = false;
                label3.Text = "k =";
                label4.Text = "";
                label5.Text = "";
                textBox2.Visible = false;
                textBox3.Visible = false;
            }
        }
        private void UpdateListView()
        {
            bool linear = checkBox1.Checked;
            bool parabola = checkBox2.Checked;
            bool hyperbola = checkBox3.Checked;

            int count = manager.MathFunctions.Count;

            listView1.Items.Clear();

            for (int i = 0; i < count; i++)
            {
                OneArgument function = manager.MathFunctions[i];
                if (linear && function is Linear ||
                    parabola && function is Parabola ||
                    hyperbola && function is Hyperbola)
                {
                    string[] row = { i.ToString(), function.name, function.Show(),
function.argument.ToString(), String.Format("{0:f2}", function.value) };
                    listView1.Items.Add(new ListViewItem(row));
                }
            }
            button3.Enabled = listView1.SelectedItems.Count > 0 &&
listView1.Items.Count > 0;
            if (listView1.Items.Count > 10)
            {
                columnHeader4.Width = 90;
            }
            else
            {
                columnHeader4.Width = 115;
            }
        }
        private void listView1_ColumnWidthChanging(object sender,
ColumnWidthChangingEventArgs e)
        {
            e.Cancel = true;
            e.NewWidth = listView1.Columns[e.ColumnIndex].Width;
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            resetForm();
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            int index = comboBox1.SelectedIndex;
            int num = manager.getNumParams(index);
            double argument;
            double[] param = new double[num];
            string[] text = new string[numTypes];

            text[0] = textBox1.Text.Replace('.', ',');
            text[1] = textBox2.Text.Replace('.', ',');
            text[2] = textBox3.Text.Replace('.', ',');

            for (int i = 0; i < num; i++)
            {
```

```
                if (!double.TryParse(text[i], out param[i]))
                {
                    MessageBox.Show(messageError, captionError, MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    return;
                }
            }

            string arg = textBox4.Text.Replace('.',',');

            if(!double.TryParse(arg, out argument))
            {
                 MessageBox.Show("Вы не ввели аргумент!", captionError,
MessageBoxButtons.OK, MessageBoxIcon.Information);
                 return;
            }

            OneArgument function;

            if (index == 0)
            {
                function = new Linear();
            }
            else if (index == 1)
            {
                function = new Parabola();
            }
            else
            {
                function = new Hyperbola();
            }

            try
            {
                function.Parameters(param);
                function.Calculate(argument);
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message, captionError, MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }

            manager.MathFunctions.Add(function);
            UpdateListView();
            resetForm();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            resetForm();
        }

        private void listView1_SelectedIndexChanged(object sender, EventArgs e)
        {
            button3.Enabled = listView1.SelectedItems.Count > 0 &&
listView1.Items.Count > 0;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            int id = 0;
            int ord = 0;
            foreach (ListViewItem eachItem in listView1.SelectedItems)
            {
                id = int.Parse(eachItem.SubItems[0].Text);
                ord = eachItem.Index;
                manager.MathFunctions.RemoveAt(id);
                UpdateListView();
```

```
            }

            int count = listView1.Items.Count;

            if (count > 0)
            {
                listView1.Items[(count > ord ? ord : ord - 1)].Selected = true;
                listView1.Select();
            }
        }

        private void listView1_ColumnClick(object sender, ColumnClickEventArgs e)
        {
            if (listView1.Items.Count == 0)
                return;

            if(e.Column == 4)
            {
                manager.SortByValue();
                UpdateListView();
            }
            else
            {
                MessageBox.Show("Доступна только сортировка по значению!",
captionError, MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            UpdateListView();
        }

        private void checkBox2_CheckedChanged(object sender, EventArgs e)
        {
            UpdateListView();
        }

        private void checkBox3_CheckedChanged(object sender, EventArgs e)
        {
            UpdateListView();
        }

        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "txt files (*.txt)|*.txt";
            saveFileDialog1.FilterIndex = 2;
            saveFileDialog1.RestoreDirectory = true;
            saveFileDialog1.Title = "Создать";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                currentFile = saveFileDialog1.FileName;
                manager.CreateFile(currentFile);
                string defaultFile = Path.GetFileName(currentFile);
                this.Text = defaultFile + " - " + title;
                UpdateListView();
                resetForm();
            }
        }

        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            manager.SaveFile(currentFile);
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            BinaryFormatter formatter = new BinaryFormatter();
            OpenFileDialog openFileDialog1 = new OpenFileDialog();
```

```
            openFileDialog1.Filter = "Text Files (.txt)|*.txt|All Files (*.*)|*.*";
            openFileDialog1.FilterIndex = 1;
            openFileDialog1.Multiselect = false;
            openFileDialog1.Title = "Открыть";
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                currentFile = openFileDialog1.FileName;
                manager.OpenFile(currentFile);
                UpdateListView();
                string defaultFile = Path.GetFileName(currentFile);
                this.Text = defaultFile + " - " + title;
            }
        }

        private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "txt files (*.txt)|*.txt";
            saveFileDialog1.FilterIndex = 2;
            saveFileDialog1.RestoreDirectory = true;
            saveFileDialog1.Title = "Сохранить как...";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                currentFile = saveFileDialog1.FileName;
                manager.SaveFileAs(currentFile);
                string defaultFile = Path.GetFileName(currentFile);
                this.Text = defaultFile + " - " + title;
            }
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Программа \"Менеджер математических функций\".
\r\nРазработал студент группы 171ПИ Сероусов Виталий.", "О программе",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

7. Список литературы

http://msdn.microsoft.com

http://stackoverflow.com